REMARKS

This is filed in response to the Office Action dated December 13, 2007, rejecting claims 1 – 66 under 35 U.S.C. § 103. In view of the the remarks that follow, the Applicant submits that all pending claims are in condition for allowance.

Claims 64 – 66 are cancelled without prejudice.

I. Rejection of Claims 1 – 63 Under 35 U.S.C. § 103

A. Independent Claim 1

Claim 1 stands rejected under 35 U.S.C. § 103(a) as being allegedly unpatentable over Brown, III et al., US 6,240,508, ("Brown") in view of Jagannathan et al., US 5,692,508 ("Jagannathan") and Sekiguchi et al., US 2001/0016879 ("Sekiguchi").

Claim 1 is directed towards an embedded processor comprising a plurality of processing units that each execute processes or threads (collectively, "threads"). One or more execution units are shared by the processing units and execute instructions from the threads. An event delivery mechanism is in communications coupling with the plurality of processing units and delivers events (e.g., interrupts) to respective threads without execution of instructions by the processing units.

Neither Brown, Jagannathan, nor Sekiguchi, individually and in combination, teach or suggest an embedded processor meeting the limitations of claim 1. More specifically, Brown purports to disclose a synchronized macropipelined microprocessor chip and a CPU having several distinct "units," including an "execution unit." *See,* Brown, col. 7, lines 24 – 48.

Nowhere, however, does that publication teach or suggest an event delivery mechanism that is in communication with a plurality of processing units and that delivers events to respective threads without execution of instructions by the processing units. The Office Action does not contend otherwise — but, instead, asserts that Jagannathan and Sekiguchi remedy this deficiency.

Jagannathan purports to disclose an exception dispatcher. *See*, e.g., Jagannathan, col. 24, line 62. However, that publication fails to teach or suggest that the dispatcher could dispatch exceptions without execution of instructions by processing units, e.g., as required by claim 1 of the instant application. To the contrary, the Jagannathan dispatcher explicitly requires processor execution, and the patent goes as far as providing pseudo-code of the instructions necessary to perform the dispatching function:

```
1:    (define (exception-dispatcher type . args)
2:        (save-current-continuation)
3:        (let ((target handler (get-target&handler type args)))
4:            (cond ((eq? target (current-thread))
5:                (apply handler args))
6:                (else
7:                (signal target handler args)
8:                (case ((exception-priority type))
9:                ((continue) (return))
10:               ((immediate) (switch-to-thread target))
11:               ((reschedule) (yield-processsor)))))))
```

See Jagannathan, at col. 24, line 24, to col. 25, line 29 (emphasis added).

Jagannathan details the purpose of each line of pseudo-code, reciting, for example, that in line 7 "... the [exception] dispatcher sends the exception to the target thread (line 7). Sending a thread a signal is equivalent to interrupting the thread and pushing … the signal handler … onto the thread's stack, and resuming the thread which causes the signal handler to be executed." *See,* Jagannathan, col 25, lines 22 – 26. Therefore, dispatching an exception to a thread necessarily includes sending a signal to the signal handler, and executing that signal handler — i.e., executing processor instructions.

The signal handler purportedly disclosed in Jagannathan is an integral aspect of the exception dispatching mechanism allegedly provided by that publication. For example, as discussed above, in order to deliver an exception, the signal handler is pushed onto the thread's stack and executed. *See* Jagannathan, col. 25, lines 22 – 26. In order to have an exception dispatcher without instruction execution according to Jagannathan, the signal handler would have to be removed, and there is no reason to believe that the exception dispatcher would function without the signal handler.

In sum, Jagannathan does not remedy the deficiencies of Brown. The remarks in the Office Action are in accord with that view. *See* Office Action, mailed December 13, 2007, page 4 ("Jagannathan teaches a dispatcher...but fails to specifically teach the dispatcher as hardware or executed external to the processor."). Rather, the Office Action incorrectly asserts that Sekiguchi remedies such deficiencies. *Id.*

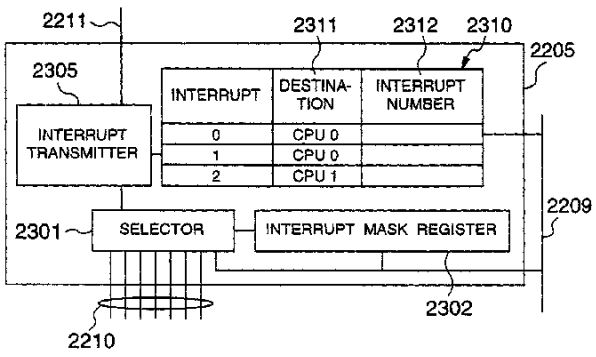
Sekiguchi purports to disclose a method and system for configuring a plurality of operating systems. *See* Sekiguchi, ¶ 0015. Nowhere does that publication teach or suggest an

event delivery mechanism that is in communication with a plurality of processing units and that delivers events to respective threads without execution of instructions by the processing units, e.g., as required by pending claim 1.

The Office Action incorrectly asserts that Sekiguchi "teaches an interrupt controller that is external to the processor and provides the functionality of the dispatcher such that events are delivered without execution of instructions by the processors." *See* Office Action, mailed December 13, 2007, pages 4 – 5. However, the Office Action unfairly attributes to the Sekiguchi controller capabilities that simply are not disclosed in that publication.

Specifically, for example, nowhere does Sekiguchi teach or suggest that the interrupt controller can deliver events to threads. At most, that publication suggests that the controller can specify a CPU destination for an interrupt — not a particular thread, process or otherwise! For example, Figure 23 of Sekiguchi, reprinted left, clearly shows that the only destination specified for an interrupt is a CPU. There is no indication of any capability for delivery to a thread, process, or otherwise.

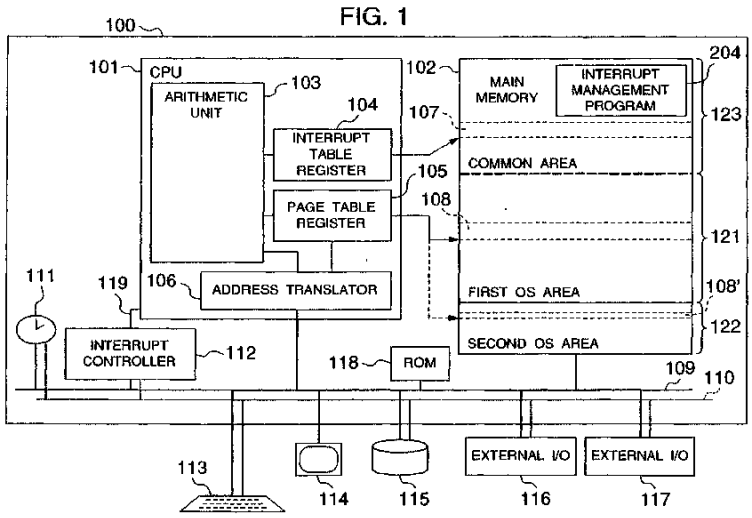
FIG. 23



Indeed, closer scrutiny of Sekiguchi reveals that processor instructions are actually required for interrupt delivery. Referring to Figure 1 of that publication, reprinted below, there is shown, among other things, an interrupt table register 104 within the CPU 101. In regard to that drawing, Sekiguchi says "[w]hen an interrupt occurs, the processor 101 receives the interrupt

number from the interrupt controller 112. By using this number as a search index, the processor acquires an interrupt handler address from the interrupt table 107 to pass the control to the interrupt handler." *See* Sekiguchi, ¶ 0050, lines 10 – 15.

FIG. 1



It is thus quit clear that Sekiguchi requires processor instructions to handle interrupts. This runs contrary to the claimed delivery mechanism which calls for delivery of events to respective threads without execution of instructions by the processing units, e.g., as recited in claim 1.

In view of the foregoing, it is evident that the combination of Brown, Jagannathan and Sekiguchi fails to teach, suggest or otherwise render unpatentable the subject matter of claim 1. The same is true for claims 2 – 6 which depend from claim 1 and recite further limitations thereon.

B. Independent Claims 7, 17, 28, 32, 35, 41, 47, 57, 61

Independent Claims 7, 17, 28, 32, 35, 41, 47, 57, 61 recite *inter alia* an event delivery mechanism that is in communication coupling with a plurality of processing units (or "virtual processing units" in claims 7, 17, 32, 41, 47 and 61) and that delivers events to respective threads with which those events are associated without execution of instructions by said processing units

(or "virtual processing units" in claims 7, 17, 32, 41, 47 and 61). For at least the reasons discussed above, Brown, Jagannathan and Sekiguchi, individually and in combination, fail to teach or suggest an event delivery mechanism with such limitations — and, thus, fail to render obvious the subject matter of claims 7, 17, 28, 32, 35, 41, 47, 57, 61.

The same is true for claims 8 – 14, which depend from claim 7 and recite further limitations thereon; claims 18 – 23, which depend from claim 17 and recite further limitations thereon; claims 29 – 31, which depend from claim 28 and recite further limitations thereon; claims 33 and 34, which depend from claim 32 and recite further limitations thereon; claims 36, 37, 39 and 40, which depend from claim 35 and recite further limitations thereon; claims 42 – 46, which depend from claim 41 and recite further limitations thereon; claims 48 – 56, which depend from claim 47 and recite further limitations thereon; claims 58 – 60, which depend from claim 57 and recite further limitations thereon; and claims 62 and 63, which depend from claim 61 and recite further limitations thereon.

C. Claims 4, 24 – 26, 38 and 53 – 55

Claims 4, 24 – 26, 38 and 53 – 55 stand rejected under 35 U.S.C. § 103(a) as allegedly unpatentable over Brown in view of Jagannathan and Sekiguchi, and further in view of Eggers (Eggers et al. "Simultaneous Multithreading: A Platform for Next-Generation Processors, IEEE, 1997; pages 12 – 19.). These claims depend from claims 1, 17, 35 or 47, respectively, and are patentably distinct from the teachings of Brown in view of Jagannathan and Sekiguchi for at least the reasons above. Insofar as Eggers fails to remedy the deficiencies of those other references — namely, failing to teach or suggest *inter alia* an event delivery mechanism that is in

communication coupling with a plurality of (virtual) processing units and that delivers events to respective threads with which those events are associated without execution of instructions by said processing units — claims 4, 24 – 26, 38 and 53 – 55 are patentably distinct from the combination of Brown, Jagannathan, Sekiguchi and Eggers.

D. Claim 15

Claim 15 stands rejected under 35 U.S.C. § 103(a) as being allegedly unpatentable over Brown in view of Jagannathan and Sekiguchi, and further in view of Gosior et al. (US 2003/0120896 A1). Claim 15 depends from claim 7 and is patentably distinct from the teachings of Brown in view of Jagannathan and Sekiguchi for at least the reasons above. Insofar as Gosior fails to remedy the deficiencies of those other references — namely, failing to teach or suggest *inter alia* an event delivery mechanism that is in communication coupling with a plurality of virtual processing units and that delivers events to respective threads with which those events are associated without execution of instructions by said processing units — claim 15 is patentably distinct from the combination of Brown, Jagannathan, Sekiguchi and Gosior.

II. Conclusion

In light of the foregoing, the Applicant believes that the application is in condition for allowance. The Examiner is encouraged to telephone the undersigned attorney for Applicants if such communication will expedite prosecution of this application.

Respectfully submitted,

Date: June 5, 2008

/David J. Powsner/

David J. Powsner (Reg. No. 31,868)
Attorney for Applicants

Nutter McClennen & Fish LLP
World Trade Center West
155 Seaport Boulevard
Boston, MA 02210-2604

Tel: (617) 439-2000
Fax: (617) 310-9000